MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Cambridge, Massachusetts
Project MAC

Artificial Intelligence Project
Memo 70

Memorandum MAC-M-165
June 25, 1964

Hash-Coding Functions of a Complex Variable

by William A. Martin

ABSTRACT

A common operation in non-numerical analysis is the comparison of symbolic mathematical expressions. Often equivalence under the algebraic and trigonometric relations can be determined with high probability by hash-coding the expressions using finite field arithmetic and then comparing the resulting hash-code numbers. The use of this scheme in a program for algebraic simplification is discussed.

I. Introduction

The elementary functions of a comples variable are those which can be expressed by the following recursive scheme. Any complex constant or variable will be called an expression; if u and v are expressions, then so are u+v, u·v, u^v, e^v, -u, and 1/u. The trigonometric and hyperbolic functions may be expressed explicitly. Because of the defining relations of the complex field and the trigonometric identities, there are infinitely many expressions for any given function. Two expressions will be said to be equivalent if they represent the same function.

Existing schemes for expression comparison use the defining relations along with some additional conditions to put each expression in a canonical form. If the canonical forms of two expressions are identical, they must represent the same function. This method has certain drawbacks. First, putting the expression in a canonical form requires the comparison of many subparts of the expression with each other. In particular, the commutative law requires that the terms in sums and products be sorted. Second, discovery that two expressions are equivalent requires a comparison of every subpart of one with the corresponding subpart of the other. Third, it is very difficult to reduce all equivalent expressions to one compact canonical form. None of the existing schemes does this.

This memo explores a probabilistic approach. Suppose $F(z) \neq G(z)$ (F and G are elementary functions), then $F(z) - G(z) = 0$ has, at most, a countable number of solutions, while the complex numbers are uncountable. Therefore, the probability that $F(z) - G(z) = 0$ for a point z chosen at randum is 0. Thus, it would be possible to test for equivalence of expressions by comparing their values at a randomly selected point. It is possible to get some approximation to this fact with the finite arithmetic

of a computer.

One method would be to substitute a random floating point for each occurrence of each distince variable and then evaluate the resulting expression using floating point arithmetic. This method is limited by overflow and roundoff error. For example, if x is a floating point number chosen at random from a flat distribution, then with probability one half x^2 is larger or smaller than all the floating point numbers; it does not appear possible to find a rule for mapping x^2 back into the floating point numbers such that the code numbers pf equivalent expressions will be very nearly the same. This overflow is difficult to avoid by restricting the initial choice of floating point numbers since expressions of the form u^v are allowed. Furthermore, if two expressions, x and y are of different orders of magnitude, then, because of roundoff error, x + y may evaluate to either x or y. This is a particular disadvantage since it is likely that an expression will be compared with subparts of itself. The same problems arise with a floating point approximation to the complox numbers.

One possible answer, which we investigate here, is to use a finite field, instead of the infinite field of real numbers.

II. Finite Fields and the Exponent Arithmetic

a. Finite Fields

The use of floating point numbers in the code number scheme is limited because the sum or product of two floating point numbers is not necessarily a floating point number. This problem is avoided if a finite field, F, is used, since the field can be chosen small enough so that every element can be represented by a computer number. The task is to choose F such that expressions which are equivalent in the complex numbers are also equivalent

in it. That is, we need a homomorphism from the complex numbers onto F. We now develop a field which meets this requirement in many, but not all, cases.

An abelian group G is a set of elements with an operation x and an identity element e such that:

1. $a \in G$, $b \in G$ then $a \times b \in G$
2. $a \in G$, then $ae = ea = a$
3. $a \in G$, then $\exists\ a^{-1} \in G \ni aa^{-1} = a^{-1}a = e$
4. $a \in G$, $b \in G$, then $ab = ba$

A finite field F is a finite set of elements with an operation + under which the elements of F form a group with identity 0 (the additive group), and an operation $\cdot$ under which the elements of $F' = F - 0$ from a group with identity 1 (the multiplicative group). In addition the relations

$$a\cdot(b + c) = a\cdot b + a\cdot c \quad \text{and} \quad a\cdot 0 = 0$$

hold.

If m and n are integers and p is a prime integer then $c = (m+n) \bmod p$ means that c equals the remainder of $(m+n)/p$. Multiplication mod p is defined similarly. It can be verified that the integers less than a prime from a finite field under the operations addition and multiplication mod p. The additive inverse of 1, -1 is seen to be p-1 since $p-1+1 = 0 \bmod p$.

b. <u>The element i</u>

In the complex field there is an element i such that $i\cdot i = -1$, so such an element is also required in F. To see how this restricts the choice of p one needs the fact found in the references that the multiplicative group F' of a finite field is cyclic. This means that there is an

element α (called a generator) in F' such that every element in F' is some power of α . In fact, F' can be written 1, α, α^2, ... , α^{p-2} and $\alpha^{p-1} = 1$. Since p is a prime it is odd and so $\frac{p-1}{2}$ is an integer. $\alpha^{(p-1)/2} = -1$ since $\alpha^{(p-1)/2} \neq 1$ and $(\alpha^{(p-1)/2})^2 = 1$. If $\frac{p-1}{2}$ is even then $r = \frac{p-1}{4}$ is an integer such that if $i = \alpha^r$, $i^2 = -1$. Note that either α^r or α^{3r} can be chosen as i and the other becomes -i. We have thus shown that F will have an element i if and only if p is of the form $4q + 1$.

c. <u>The Exponent Arithmetic</u>

In the complex numbers, one might have to test for the equivalence of two expressions such as u^{v+1} and u^v. u, where the exponent arithmetic is also performed in the complex numbers. However, since the multiplicative group is a cyclic group with one less element than F, v + 1 must be computed mod (p - 1). Since an isomorphism does not exist between the additive and multiplicative groups of F, the exponent operations cannot be perfromed in it. This failure of the finite field evaluation to be recursive in the exponent direction is a serious limitation. Furthermore, since p - 1 is not a prime the exponent operations will not form a field. Fortunately, many expressions encountered in analysis have rather simple exponents and so much can be saved by evaluating the exponents in the E arithmetic which we now define.

Let the basic elements of E be the integers less than p - 1. Addition and multiplication are mod (p - 1). It is easy to see that all elements have an additive inverse. No even integers have a multiplicative inverse, however, for this would imply:

$$2 \cdot s \cdot (2 \cdot s)^{-1} = 4 \cdot q \cdot m + 1$$

or

$$2\cdot(s\cdot(2\cdot s)^{-1} - 2\cdot q\cdot m) = 1$$

which is a contradiction since 1 has no divisors in the integers. If we take q prime, then the odd integers other than q have a multiplicative inverse as a consequence of the Euler theorem (see Ref., Albert, p.47):

Let $\phi(m)$ be the number of integers g such that $0 < g < m$ and g is prime to me. Then

$$a^{\phi(m)} = 1 \bmod m$$

for every a prime to m.

The failure of the even integers to have a multiplicative inverse means that $u^{1/2 + 1/2}$ will not evaluate to u. We thereform adjoin to the basic elements of E the element ρ, where $2\rho = 1$. Closing E under multiplication and addition would require that all the elements of the form b^{1} (b an integer less than p - 1 and b odd) be in E. However, many cases can be covered if we allow only elements a or $b\rho$.

d. Square Roots in F

u^{ρ} should evaluate to be the square root of u in F; however, only one half the elements in F have a square root, these are the even powers of the generator, α, of F'. Since $(\alpha^n)^r$ is even for even n and any r only 1/4 of the elemtns could have a square root computable by raising the element to some power, that is, by assigning some integer to ρ. However a method of finding the roots of this samller set can be found, as will be shown next, i.e., there exists a ρ such that if $u = \alpha^{4n}$, then $u^{\rho} = \alpha^{2n}$.

If p is of the form 4q + 3, then since $4n(q + 1) = 2n \bmod (4q + 2)$ n< q one finds $(\alpha^{4n})^{q+1} = \alpha^{2n}$. q+1 is thereform the proper value for ρ. The requirement that p be of the form 4q + 3 is unfortunately in conflict with the earlier requirement that p be of the form 4q + 1. By choosing

$p = 8q' + 5 = 4(2q' + 1) + 1$ one obtains by a similar argument the square root of 1/8 of the elemnts with $\rho = q' + 1$.

e. Trigonometric Identities

Define in the usual manner:

$$\sin\ \theta = \frac{e^{i\theta} - e^{-i\theta}}{2i} \qquad \sinh\ \theta = \frac{e^{\theta} - e^{-\theta}}{2}$$

$$\cos\ \theta = \frac{e^{i\theta} + e^{-i\theta}}{2} \qquad \cosh\ \theta = \frac{e^{\theta} + e^{-\theta}}{2}$$

where e is an element of F yet to be chosen. Note that $i^2 \neq -1$ in the E arithmetic but this does not arise in taking sums and products of the above functions. For instance:

$$\sin^2\theta + \cos^2\theta = \left(\frac{e^{i\theta} - e^{-i\theta}}{2}\right)^2 + \left(\frac{e^{i\theta} + e^{-i\theta}}{2}\right)^2$$

$$= \frac{e^{i\theta}\cdot e^{i\theta} - 2\cdot e^{i\theta}\cdot e^{-i\theta} + e^{-i\theta}\cdot e^{-i\theta}}{-4} + \frac{e^{i\theta}\cdot e^{i\theta} + 2e^{i\theta}\cdot e^{-i\theta} + e^{-i\theta}\cdot e^{-i\theta}}{4}$$

$$= \frac{-2\cdot 1}{-4} + \frac{2\cdot 1}{4}$$

$$= 1$$

It is necessary that $e^{i\pi} = -1$ and $e^{i\pi/2} = +i$. If e is to have a square root it must be an even power of α; taking $e = \alpha^{2n}$ one obtains for $p = 8q' + 5$:

$$e^{i\pi} = -1$$

$$\alpha^{2ni\pi} = \alpha^{4q' + 2}$$

$$2ni\pi = (4q' + 2) \bmod (8q' + 4)$$

$$ni\pi = (2q' + 1) \bmod (8q' + 4) \qquad (1)$$

For any choice of n, e is determined, providing equation (1) can be solved for the element π. Some reflection will show that trigonometric

calculations may involve roots of e greater than 2. Suppose n is chosen odd, then one square root of e, α^n, has no square root, nor does $-\alpha^n$ the other square root of e. That $-\alpha^n$ does not have a square root is a consequence of the choice of -1 as a square. $-\alpha^n = -1\ \alpha^n$ = (square)(nonsquare) and a suqare times a nonsquare must be a nonsquare. From this one can see that if e is to have a 2^mth root, it must be chosen of the form $\alpha^{c\cdot 2^m}$. Note that the choice of n divides the elements of F; between the roots and powers of e.

Assuming that square roots of e will be of chief interest, we return to the solution of equation (1) for the case n = 1. A sufficient condition for solution is that i be of the form 4r + 1, whence equation (1) yields $\pi = 2q' + 1$. Thus π is equal to q and should be taken prime.

It remains to verify that for the choices made the pairs (sin θ, cos θ) can be made to take on $4\pi = 8q' + 4$ distince pairs of values in F as θ runs over the 6π elements of E. Each pair occurs at most twice. Since i is odd, it is relatively prime to 4π unless $i = \pi$. As π is of the form $2q' + 1$ and i of the form $4n + 1$, $i \neq \pi$ if q' is odd. With i relatively prime to 4π the relation $y = ix \bmod 4\pi$ has a unique solution $x = i^{-1}y$ for each x less than 4π. The 6π elements of E can be represented in the form x where x takes on the odd values 0 to $4\pi - 1$ and all the values from 4π to $8\pi - 1$. Then $e^{ix} = \alpha^{ix} = \alpha^y$ takes on all the values in F' and no value more than twice. It is easy to show that if

$$a_1 = \frac{x_1 + 1/x_1}{2} \qquad b_1 = \frac{x_1 - 1/x_1}{2i}$$

and

$$a_1 = \frac{x_2 + 1/x_2}{2} \qquad b_1 = \frac{x_2 - 1/x_2}{2i}$$

then $x_1 = x_2$, which completes the proof.

f. Summary of the Requirements

1. p of the form $8q' + 5$.

2. q' odd as a sufficient condition.

3. i of the form $4m + 1$ as a sufficient condition.

4. $2q' + 1$ prime

III. Machine Realization--Finding a Prime

The requirements in section II can be rephrased as:

1. $p = 16n + 13$ prime.

2. $\pi = 4n + 3$ prime.

3. i of the form $4m + 1$.

Another requirement is:

4. p less than 1/2 the largest machine integer.

This allows addition without overflow. The multiplicative inverse of an element a in F is found by noting that $a^{-1} = a^{p-2}$. To raise an element a to any power we begin multiplying it by itself, creating the numbers $b_i = a^{2^i}$, we then express the power as a binary number and add up the appropriate b_i. This leads to the requirement:

5. p - 2 should be expressed as a few powers of 2.

To find a prime for the 7094 the following procedure was followed:

1. Beginning with $n = 2^{29}$ test if $p = 16n + 13$ is prime by dividing by every odd number up to $\sqrt{p}$.

2. Test if $4n + 3$ is prime.

3. Find a generator α_1 of F'. (If a is not a generator then $a^2 = 1$ or $a^4 = 1$ or $a^{\pi} = 1$ or $a^{2\pi} = 1$. Raise a to these four powers by the scheme above.) Almost 1/2 of the elements are generators so one is quickly found.

4. Compute $i_1 = \alpha_1^{4n+3}$.

5. If i_1 is not odd compute $\alpha_2 = \alpha_1^{-1}$, and $i_2 = \alpha_2^{4n+3}$. i_2 will be odd.

6. Check to see if this odd i is of the form 4n + 1.

After two hours of computation this procedure resulted in:

p = 8,589,949,373

α = 13,560, 097

e = 8,364,320,344

π = 2,147,487,343

i = 5,525,736,173

Machine language routines were written to perform the operations in the E and F arithmetic. The evaluation of a code number is performed in the LISP language. The following conventions were followed:

1. Multiplication by φ is indicated with the computer word minus sign.

2. Recursion in the exponent direction follows the pattern F, E, F, E, etc.

3. Floating point numbers in an expression are treated as rational numbers and the corresponding F or E element is computed.

4. Whenever the element φ^2 appears, it is changed to φ by multiplying by the integer assigned to φ.

IV. The Probability of Error

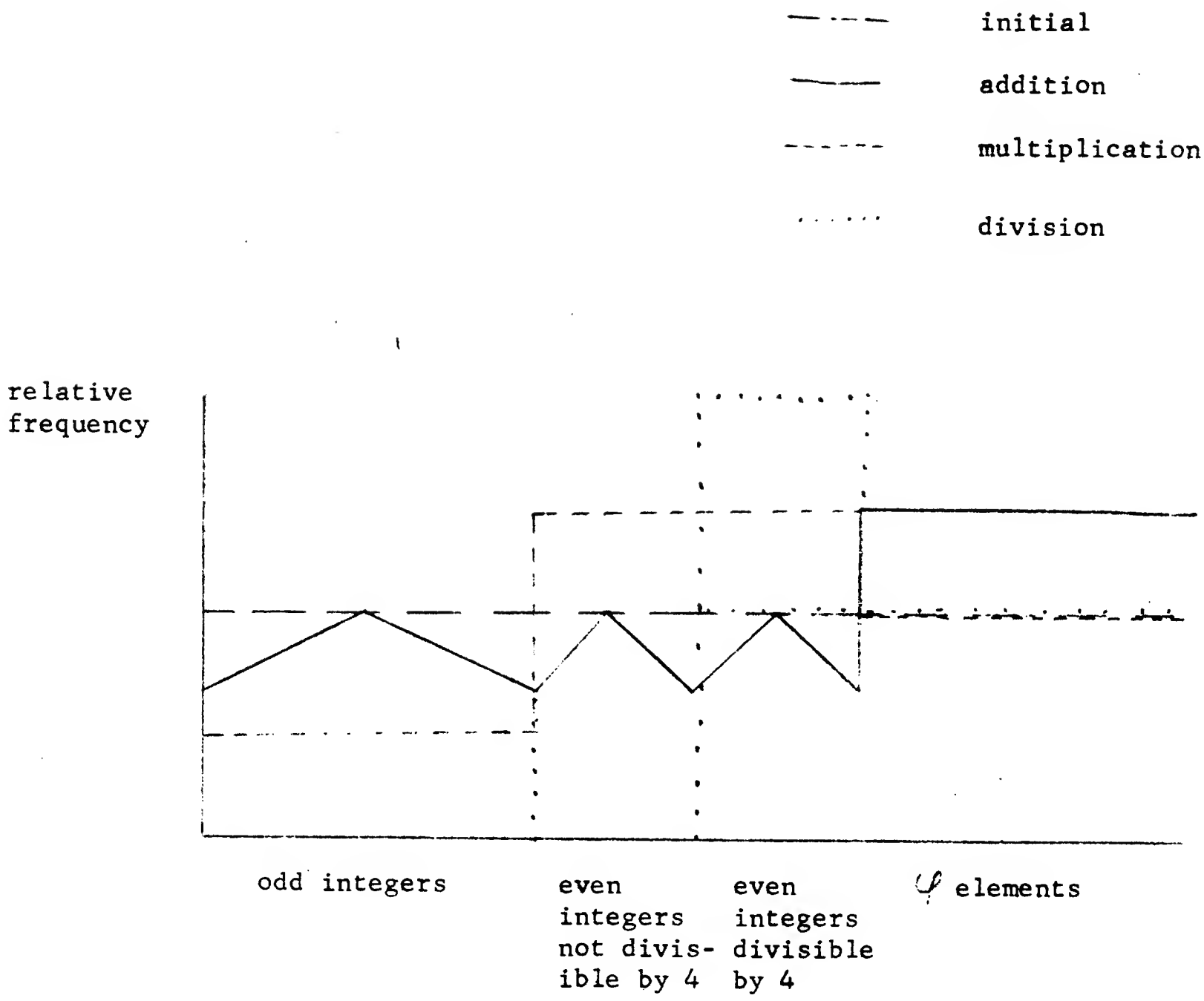
Estimation of the probability of error is difficult. The average probability of error for certain subsets of expressions will differ from that for all expressions. No statistics are available on the expressions which will be encountered in practice.

It is possible that two expressions which represent the same function will receive different code numbers because some exponent operation does

not preserve the equivalence. Study of section II should make clear under what circumstances this will happen.

In the simplification program described in the next section, expressions with the same code number are considered equivalent. Therefore, an accidental match of non-equivalent expressions is very serious. If we could show that the operations in the E and F arithmetic mapped their sets of elements uniformly back onto themselves, then the probability of a match between two expressions selected at random from the set of all expressions would be 1/p. Unfortunately, this is not the case. In the F arithmetic the operations of multiplication and addition and their inverses do satisfy this criterion. Looking at the cyclic group F', however, one sees that raising any element in F except a multiple of π to all powers will produce either 1/4, 1/2, or all the elements of F. Thus, exponentiation tends to map the elements into the 4th powers of a generator and so increase the probability of random match.

The same bunching occurs in the E arithmetic. The distribution of elements after n operations can only be found using a rather complicated two dimensional convolution. The distributions after one operation shown in Figure 1 indicate that for moderately complicated exponents the probability of error should remain in control.



Distribution of elements in E after one operation. Within each classification, the elements are ordered in the normal manner.

Figure 1

V. The Simplification Program

A simplification program has been written in LISP. The program collects terms in sums and products, removes unnecessary levels of parentheses, and recognizes identities involving 0 and 1. The explicit operations of division and subtraction have been removed from the LISP representation, instead addition of a negative quantity and exponentiation to a negative power are used. An attempt was made to exploit the similarity between the operations on the additive and the multiplicative groups. Some further improvement could be made. The program alters list structure so that common subexpressions are simplified only once.

To recognize equivalent subexpressions the program uses the hash code just described; the hash code numbers are added to the front of the expressions. It would have been possible for all subexpressions to have hash code numbers on their property lists, however, because of limited storage the decision was made to retain only the numbers at the current level. This means that in certain situations it is necessary to recompute the numbers. The program appears to be faster and more powerful than those using canonical ordering.

References

G. Birkhoff and S. MacLane, A Survey of Modern Algebra, Macmillan, 1960.

A. Adrian Albert, Fundamental Concepts of Higher Algebra, Phoenix Science Series, University of Chicago Press, 1963.

T. Bartee and D. Schneider, "Computation with finite fields," Information and Control, 6, 1963.

H. L. Garner, "The residue number system," IRE Trans. on Electronic Computers, 8, June 1959.

Aviezri S. Fraenkel, "The use of index calculus and Mersenne primes for design of a high-speed digital multiplier," JACM, January 1961.

Dean Wooldridge, Jr., "An algebraic simplify program in LISP," Stanford Artificial Intelligence Project, Memo No. 11, December 1963.

N. C. Ankeny, MIT Mathematics Department, discussion.